What’s an algorithm?

“A procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation.” — webster.com

“An algorithm is a finite, definite, effective procedure, with some input and some output.” — Donald Knuth
What is this course about?

• In CS230, you learnt how to:
  • Abstract functionality from design
  • Design efficient data structures
  • Design modular applications

• You used to,
  • Code all of that
  • Test it to see if it works
  • Then, the fun starts 😊

What is this course about?

• In CS231, you learn the design and analysis of algorithms to solve problems.

• We’ll always focus on three things:
  • How to understand and define a problem
  • How to implement an algorithm that “correctly” solves that problem
  • How to analyze the performance of that algorithm

• The goal is to define correct and efficient algorithms without having to implement and test them.

Remember: You cannot prove correctness by example!
Why take CS231?

Because ... It’s a major requirement!

Really, why?

- Understanding and Remembering:
  - Recognize algorithmic techniques used to solve a problem.
  - Identify the correctness, or lack thereof, of an algorithm.
- Critical Thinking:
  - Dissect new problems to identify their input and corresponding output.
- Practical Thinking:
  - Determine appropriate algorithmic techniques to solve new problems, by relating new problems to ones in their foundation knowledge.
  - Define correct algorithms to solve new problem, and prove their correctness.
  - Critique existing algorithms.
  - Calculate the asymptotic run time complexity of new algorithms.
- Projects and Research:
  - Coordinate tasks and collaborate on writing a final paper.
  - Identify high quality scholarly articles, and their contributions.
  - Summarize existing algorithmic research on a topic of their choice
  - Present summary of research to peers, as part of a team.
- Interpersonal Relationships:
  - Collaborate with peers on dissecting new problems.
  - Give feedback to peers on their proofs.
  - Take responsibility for work performed as part of a group.
Textbook


Assignments

- **Schedule**
  - Posted on Thursday / Friday
  - Due the following Thursday (in class)

- **Submission**
  - Write your assignments in latex, try sharelatex.com
  - You will be provided with a Latex template every assignment
    - Template must be used as is
  - Bring hard copy of assignment to class on Friday
  - Upload soft copy on your CS231 assignment directory

- **Proof modules**
  - In most assignments, you will find a problem marked with [Proof-problem]
  - For these problems, you need to carefully formulate and write your arguments for the correctness of your solutions.
More on proof modules

- Type the proof using latex
- Submit on the same day as the assignment, but in a separate sheet
- This proof will be graded through the weekend, and you’ll get feedback on it by Monday
- Grading:
  - If the proof is correct and complete, you get full points
  - If edits need to be made, you can resubmit the following Friday
  - Note: you can only resubmit once

More ...

- You’ll have three exams during the semester,
  - 1st exam on Oct 1
  - 2nd exam on Nov 5
  - 3rd exam on Dec 3
- All exams are in-class and open book
- There will be no final exam.
- Instead, there will be a final short paper and presentation.
CS231 course webpage

- It contains all course info
  - The schedule for the semester
  - Office hours discussion sections and help room
  - The course syllabus

- Make sure you check it often

- Let’s take a quick look

Stable matching problem

Our first algorithm 😊
Matching Residents to Hospitals

• **Goal.** Given a set of preferences among hospitals and medical school students, design a self-reinforcing admissions process.

• **Unstable pair:** applicant x and hospital y are *unstable* if:
  • x prefers y to its assigned hospital.
  • y prefers x to one of its admitted residents.

• **Stable assignment.** Assignment with no unstable pairs.
  • Natural and desirable condition.
  • Individual self-interest will prevent any applicant/hospital deal from being made.

Stable Matching Problem

• **Input:**
  • Given n residents and n hospitals, with their rating of each other.
  • Each resident lists hospitals in order of preference from best to worst.
  • Each hospital lists residents in order of preference from best to worst.

• **Goal:**
  • Find a "suitable" matching.

<table>
<thead>
<tr>
<th>Residents' Preference Profile</th>
<th>Hospitals' Preference Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>favorite</strong></td>
<td><strong>least favorite</strong></td>
</tr>
<tr>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>Xavier</td>
<td>Amy</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
</tr>
<tr>
<td>Zeus</td>
<td>Amy</td>
</tr>
</tbody>
</table>
Let’s formalize this ...

- M =
- W =
- M x W:
- S:

Some definitions ...

- **Perfect matching**: everyone is matched monogamously.
  - Each resident gets exactly one hospital.
  - Each hospital gets exactly one resident.

- **Stability**: no incentive for some pair of participants to undermine assignment by joint action.
  - In matching M, an unmatched pair r-h is **unstable** if ...
  - Unstable pair r-h could each improve by breaking contracts.

- **Stable matching**: perfect matching with no unstable pairs.

- **Stable matching problem**: Given the preference lists of n residents and n hospitals, find a stable matching if one exists.
Stable Matching Problem

• Q. Is assignment X-C, Y-B, Z-A stable?

  Residents’ Preference Profile

<table>
<thead>
<tr>
<th>Favorite</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
</tr>
<tr>
<td>Zeus</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
</tbody>
</table>

  Hospitals’ Preference Profile

<table>
<thead>
<tr>
<th>Favorite</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Zeus</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
</tbody>
</table>

• A.

  An unstable pair (r,h) could each improve by joint action.

• Q. Is assignment X-C, Y-B, Z-A stable?

  Residents’ Preference Profile

<table>
<thead>
<tr>
<th>Favorite</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
</tr>
<tr>
<td>Zeus</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
</tbody>
</table>

  Hospitals’ Preference Profile

<table>
<thead>
<tr>
<th>Favorite</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Zeus</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
</tbody>
</table>
Stable Matching Problem

- Q. Is assignment X-A, Y-B, Z-C stable?
- A. Yes.

<table>
<thead>
<tr>
<th>Residents’ Preference Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
</tr>
<tr>
<td>Xavier</td>
</tr>
<tr>
<td>Yancey</td>
</tr>
<tr>
<td>Zeus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hospitals’ Preference Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
</tr>
<tr>
<td>Amy</td>
</tr>
<tr>
<td>Bertha</td>
</tr>
<tr>
<td>Clare</td>
</tr>
</tbody>
</table>

Stable Roommate Problem

- Q. Do stable matchings always exist?
- A. Not obvious a priori.

- Stable roommate problem.
  - 2n people; each person ranks others from 1 to 2n-1.
  - Assign roommate pairs so that no unstable pairs.

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Bob</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>Chris</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Doofus</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

- Observation. Stable matchings do not always exist for stable roommate problem.
Propose-And-Reject Algorithm

- The [Gale-Shapley 1962] deferred acceptance algorithm is an intuitive method that guarantees to find a stable matching.

---

2012 Nobel Prize in Economics

**Lloyd Shapley.** Stable matching theory and Gale–Shapley algorithm.

*Original applications: college admissions and "traditional marriage"*

**Alvin Roth.** Applied Gale–Shapley to matching med-school students with hospitals, students with schools, and organ donors with patients.
Proof of Correctness: Termination

- Observation 1. Men propose to women in decreasing order of preference.

- Observation 2. Once a woman is matched, she never becomes unmatched; she only "trades up."

- Claim. Algorithm terminates after at most $n^2$ iterations of while loop.
- Pf. Each time through the while loop a man proposes to a new woman. There are only $n^2$ possible proposals.

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viktor</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>Wyatt</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>Kaiser</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>A</td>
</tr>
<tr>
<td>Vanya</td>
<td>D</td>
<td>E</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Zeke</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

Amy | W | X | Y | Z | V |
Bertha | Y | Z | V | W | X |
Clare | Y | Z | W | X | V |
Diana | Z | V | W | X | Y |
Erika | V | W | Y | Z | V |

$n(n-1)+1$ proposals required

Summary

- Stable matching problem.
  - Given $n$ men and $n$ women, and their preferences, find a stable matching if one exists.

- Gale-Shapley algorithm.

- Q. Does it correctly find a stable matching for any problem instance?

- Q. How to implement GS algorithm efficiently?

- Q. If there are multiple stable matchings, which one does GS find?